LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 5

CONNECT TO THE INTERNET

Oleh:

Ririn Citra Lestari NIM. 2310817120012

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 5

Laporan Praktikum Pemrograman Mobile Modul 3: Connect To The Internet ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Ririn Citra Lestari
NIM : 2310817120012

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 201903 01 011

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:

a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.

b. Gunakan KotlinX Serialization sebagai library JSON.

c. Gunakan library seperti Coil atau Glide untuk image loading. d. API yang digunakan pada modul ini adalah The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: https://developer.themoviedb.org/docs/getting-started

e. Implementasikan konsep data persistence (aplikasi menyimpan data walau pengguna keluar dari aplikasi) dengan SharedPreferences untuk menyimpan data ringan (seperti pengaturan aplikasi) dan Room untuk data relasional.

f. Gunakan caching strategy pada Room. Dibebaskan untuk memilih caching strategy yang sesuai, dan sertakan penjelasan kenapa menggunakan caching strategy tersebut.

g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose. Aplikasi harus mempertahankan fitur-fitur yang dibuat pada modul sebelumnya

Buatlah sebuah aplikasi berbasis web sederhana yang dapat melakukan operasi CRUD (Create, Read, Update, Delete) dari hasil implementasi desain basis data yang diberikan. Adapun ketentuan pembuatannya sebagai berikut:

1. Koneksi database dibuat menjadi satu file sendiri yaitu Koneksi.php, kemudian gunakan fungsi *require* ketika ingin melakukan operasi ke basis data.
2. Operasi data seperti Insert, Update, Delete, Get Data dibuat menjadi fungsi sendiri masing- masing dan disimpan di dalam satu file khusus yaitu Model.php

A. Source Code

MainActivity.kt

```
package com.example.listfilm

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.compose.rememberNavController
import com.example.listfilm.repository.MovieRepository
import com.example.listfilm.ui.screen.AppNavGraph
import com.example.listfilm.viewmodel.MovieViewModel
import com.example.listfilm.viewmodel.MovieViewModelFactory
import timber.log.Timber

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Timber.d("MainActivity onCreate() called")

        setContent {
            val navController = rememberNavController()
            val context = applicationContext
            val repository = MovieRepository(context)
            val viewModel: MovieViewModel = viewModel(
                factory = MovieViewModelFactory(repository)
            )
            AppNavGraph(navController = navController, viewModel =
viewModel)
        }
    }
}
```

Tabel 1.Source Code Soal 1 Modul 5

MovieDao.kt

```
package com.example.listfilm.data

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query

@Dao
interface MovieDao {
    @Query("SELECT * FROM movies")
    suspend fun getAllMovies(): List<MovieEntity>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(movies: List<MovieEntity>)
}
```

Tabel 2.Source Code Soal 1 Modul 5

MovieDatabase.kt

```
package com.example.listfilm.data

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [MovieEntity::class], version = 1)
abstract class MovieDatabase : RoomDatabase() {
    abstract fun movieDao(): MovieDao

    companion object {
        @Volatile private var INSTANCE: MovieDatabase? = null

        fun getDatabase(context: Context): MovieDatabase {
            return INSTANCE ?: synchronized(this) {
                Room.databaseBuilder(
                    context.applicationContext,
                    MovieDatabase::class.java,
                    "movie_database"
                ).build().also { INSTANCE = it }
            }
        }
    }
}
```

Tabel 3.Source Code Soal 1 Modul 5

MovieEntity.kt

```
package com.example.listfilm.data

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "movies")
data class MovieEntity(
    @PrimaryKey val id: Int,
    val title: String,
    val posterPath: String,
    val overview: String
)
```

Tabel 4.Source Code Soal 1 Modul 5

FilmListScreen.kt

```
package com.example.listfilm.ui.screen

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
```

```
import androidx.compose.material3.Card
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import coil.compose.rememberAsyncImagePainter
import com.example.listfilm.model.Movie
import com.example.listfilm.util.Constants
import com.example.listfilm.viewmodel.MovieViewModel

@Composable
fun FilmListScreen(navController: NavHostController, viewModel:
MovieViewModel) {
    val context = LocalContext.current
    val movies = viewModel.movies.collectAsState().value

    LazyColumn(contentPadding = PaddingValues(16.dp)) {
        items(movies) { movie: Movie ->
            Card(
                shape = RoundedCornerShape(16.dp),
                modifier = Modifier
                    .padding(bottom = 16.dp)
                    .fillMaxWidth()
            ) {
                Column(Modifier.padding(16.dp)) {
                    Image(
                        painter =
rememberAsyncImagePainter("${Constants.IMAGE_BASE_URL}${movie.posterPath}"),
                        contentDescription = null,
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp)
                            .clip(RoundedCornerShape(12.dp)),
                        contentScale = ContentScale.Crop
                    )

                    Spacer(Modifier.height(8.dp))
                    Text(text = movie.title, fontWeight = FontWeight.Bold)
                    Text(text = movie.overview, maxLines = 3)

                    Spacer(Modifier.height(8.dp))
                    Row(
                        modifier = Modifier.fillMaxWidth(),
                        horizontalArrangement = Arrangement.SpaceBetween
                    ) {
                        Button(onClick = {
                            viewModel.logOpenSiteClicked(movie)
                            val intent = Intent(
                                Intent.ACTION_VIEW,

Uri.parse("https://www.themoviedb.org/movie/${movie.id}")
                            )
```

```
                            context.startActivity(intent)
                        }) {
                            Text("Open Site")
                        }

                        Button(onClick = {
                            viewModel.logDetailClicked(movie)
                            navController.navigate("detail/${movie.id}")
                        }) {
                            Text("Detail")
                        }
                    }
                }
            }
        }
    }
}
```

Tabel 5.Source Code Soal 1 Modul 5

AppNavGraph.kt

```
package com.example.listfilm.ui.screen

import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Modifier
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import com.example.listfilm.viewmodel.MovieViewModel

@Composable
fun AppNavGraph(navController: NavHostController, viewModel:
MovieViewModel) {
    val movies = viewModel.movies.collectAsState().value

    NavHost(navController = navController, startDestination = "list") {
        composable("list") {
            FilmListScreen(navController = navController, viewModel =
viewModel)
        }
        composable("detail/{id}") { backStackEntry ->
            val id =
backStackEntry.arguments?.getString("id")?.toIntOrNull()
            val movie = movies.find { it.id == id }
            if (movie != null) {
                MovieDetailScreen(movie = movie, navController =
navController)
            } else {
                Text("Movie not found", modifier =
Modifier.fillMaxSize())
            }
        }
```

```
    }
}
```

MovieDetailScreen.kt

```
package com.example.listfilm.ui.screen

import androidx.compose.foundation.layout.*
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import com.example.listfilm.model.Movie

@Composable
fun MovieDetailScreen(movie: Movie, navController: NavHostController) {
    Column(modifier = Modifier
        .fillMaxSize()
        .padding(16.dp)) {

        Text(text = movie.title, style =
MaterialTheme.typography.headlineMedium)
        Spacer(modifier = Modifier.height(8.dp))
        Text(text = movie.overview)
        Spacer(modifier = Modifier.height(16.dp))

        androidx.compose.material3.Button(onClick = {
            navController.popBackStack()
        }) {
            Text("Back")
        }
    }
}
```

Tabel 6.Source Code Soal 1 Modul 5

MovieViewModel.kt

```
package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.listfilm.model.Movie
import com.example.listfilm.repository.MovieRepository
import com.example.listfilm.util.Constants
import kotlinx.coroutines.flow.SharingStarted
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.stateIn
import timber.log.Timber

class MovieViewModel(private val repository: MovieRepository) :
ViewModel() {
    val movies: StateFlow<List<Movie>> =
        repository.getPopularMovies(Constants.API_KEY)
```

```
            .stateIn(viewModelScope, SharingStarted.Eagerly,
emptyList())

    fun logDetailClicked(movie: Movie) {
        Timber.d("Detail clicked: ${movie.title}")
    }

    fun logOpenSiteClicked(movie: Movie) {
        Timber.d("Open site clicked:
https://www.themoviedb.org/movie/${movie.id}")
    }
}
```

Tabel 7.Source Code Soal 1 Modul 5

ListFilmApp.kt

```
package com.example.listfilm

import android.app.Application
import timber.log.Timber

class ListFilmApp : Application() {
    override fun onCreate() {
        super.onCreate()
        Timber.plant(Timber.DebugTree())
    }
}
```

Tabel 8.Source Code Soal 1 Modul 5

MovieViewModelFactory.kt

```
package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.listfilm.repository.MovieRepository

class MovieViewModelFactory(
    private val repository: MovieRepository
) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(MovieViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return MovieViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

Tabel 9.Source Code Soal 1 Modul 5

Constants.kt

```
package com.example.listfilm.util

object Constants {
    const val BASE_URL = "https://api.themoviedb.org/3/"
```

```
    const val IMAGE_BASE_URL = "https://image.tmdb.org/t/p/w500"
    const val API_KEY = "8daab9d30b9168bb81cb48d5fcd1f302"
}
```

Tabel 10.Source Code Soal 1 Modul 5

MovieRespository.kt

```
package com.example.listfilm.repository

import android.content.Context
import com.example.listfilm.data.MovieDatabase
import com.example.listfilm.data.MovieEntity
import com.example.listfilm.model.Movie
import com.example.listfilm.network.RetrofitInstance
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow
import timber.log.Timber

class MovieRepository(context: Context) {
    private val movieDao =
MovieDatabase.getDatabase(context).movieDao()

    fun getPopularMovies(apiKey: String): Flow<List<Movie>> = flow {
        try {
            val response =
RetrofitInstance.api.getPopularMovies(apiKey)
            Timber.d("Ambil dari API berhasil:
${response.results.size}")
            val movieEntities = response.results.map {
                MovieEntity(it.id, it.title, it.posterPath,
it.overview)
            }
            movieDao.insertAll(movieEntities)
            emit(response.results) // from API
        } catch (e: Exception) {
            Timber.e("Gagal ambil dari API: ${e.message}")
            val cached = movieDao.getAllMovies().map {
                Movie(it.id, it.title, it.posterPath, it.overview)
            }
            emit(cached) // fallback to local Room
        }
    }
}
```

Tabel 11.Source Code Soal 1 Modul 5

PreferenceManager.kt

```
package com.example.listfilm.data.preferences

import android.content.Context
import android.content.SharedPreferences

class PreferenceManager(context: Context) {
    private val prefs: SharedPreferences =
        context.getSharedPreferences("user_prefs",
```

```
Context.MODE_PRIVATE)

    fun setDarkMode(enabled: Boolean) {
        prefs.edit().putBoolean("dark_mode", enabled).apply()
    }

    fun isDarkMode(): Boolean = prefs.getBoolean("dark_mode", false)
}
```

Tabel 12.Source Code Soal 1 Modul 5

Movie.kt

```
package com.example.listfilm.model

import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

@Serializable
data class Movie(
    val id: Int,
    val title: String,
    @SerialName("poster_path") val posterPath: String,
    val overview: String
)
```

Tabel 13.Source Code Soal 1 Modul 5

MovieResponse.kt

```
package com.example.listfilm.model

import kotlinx.serialization.Serializable

@Serializable
data class MovieResponse(
    val page: Int,
    val results: List<Movie>
)
```

Tabel 14.Source Code Soal 1 Modul 5

RetrofitInstance.kt

```
package com.example.listfilm.network

import
com.jakewharton.retrofit2.converter.kotlinx.serialization.asConverterFactory
import kotlinx.serialization.json.Json
import okhttp3.MediaType.Companion.toMediaType
import retrofit2.Retrofit

object RetrofitInstance {
    private val json = Json { ignoreUnknownKeys = true }
    private val contentType = "application/json".toMediaType()

    val api: TmdbApiService by lazy {
        Retrofit.Builder()
            .baseUrl("https://api.themoviedb.org/3/")
```

```
            .addConverterFactory(json.asConverterFactory(contentType))
            .build()
            .create(TmdbApiService::class.java)
    }
}
```

Tabel 15.Source Code Soal 1 Modul 5

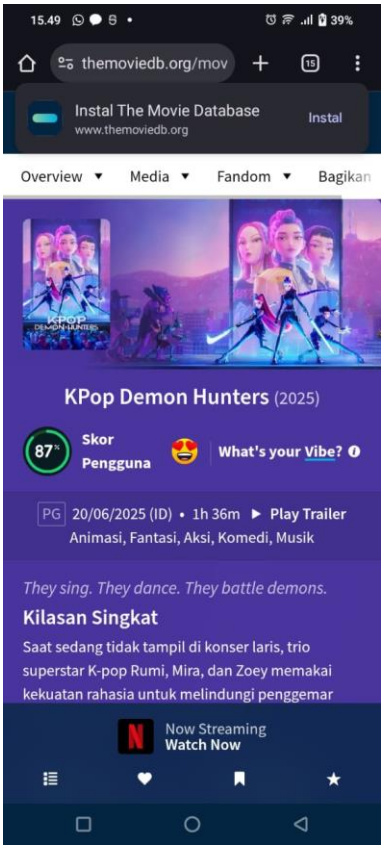
TmdbApiService.kt

```
package com.example.listfilm.network

import com.example.listfilm.model.MovieResponse
import retrofit2.http.GET
import retrofit2.http.Query

interface TmdbApiService {
    @GET("movie/popular")
    suspend fun getPopularMovies(
        @Query("api_key") apiKey: String
    ): MovieResponse
}
```

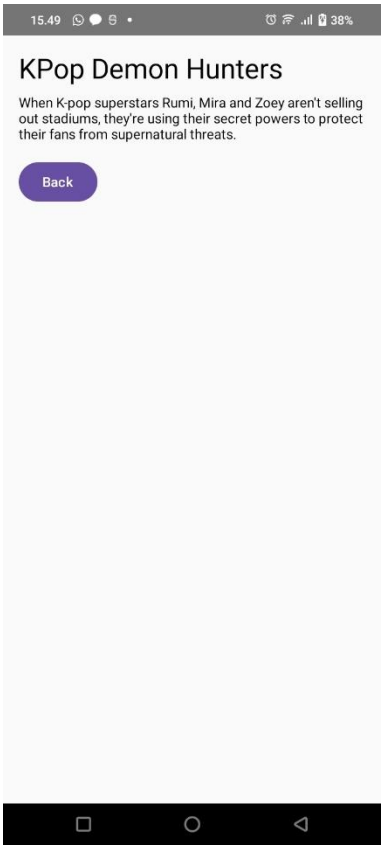
Tabel 16.Source Code Soal 1 Modul 5

B. Output Program

Gambar 1.Screenshot Hasil Jawaban Soal 1 Modul 5



Gambar 2.Screenshot Hasil Jawaban Soal 1 Modul 5



Gambar 3.Screenshot Hasil Jawaban Soal 1 Modul

C. Pembahasan

Aplikasi ListFilm merupakan aplikasi Android berbasis Jetpack Compose yang dibangun dengan arsitektur MVVM (Model-View-ViewModel)**.** Titik awal aplikasi berada di MainActivity, di mana UI Compose diinisialisasi menggunakan setContent. Di dalamnya, NavController dibuat untuk navigasi antar layar, MovieRepository diinstansiasi, dan MovieViewModel disiapkan melalui MovieViewModelFactory. ViewModel ini bertugas mengelola data dan logika bisnis aplikasi, termasuk mengambil data dari repository serta mencatat aktivitas pengguna melalui Timber. Repository sendiri menjadi jembatan antara data yang berasal dari API (TheMovieDB) dan database lokal yang dikelola dengan Room. Saat data berhasil diambil dari API menggunakan Retrofit, data tersebut disimpan ke database lokal. Jika pengambilan data dari API gagal, repository akan menampilkan data dari database lokal sebagai fallback.

Untuk kebutuhan pengambilan data, digunakan Retrofit yang dikonfigurasi melalui RetrofitInstance, dengan dukungan KotlinX Serialization untuk parsing JSON. Endpoint API didefinisikan dalam TmdbApiService. Model data dari API disimpan dalam kelas Movie dan MovieResponse, yang masing-masing dilengkapi anotasi @Serializable. Sementara itu, untuk penyimpanan lokal, digunakan entitas Room MovieEntity yang diakses melalui MovieDao, dan dikelola oleh MovieDatabase yang menerapkan pola singleton. Data ini digunakan oleh ViewModel dan disalurkan ke UI melalui StateFlow.

Navigasi antar layar ditangani oleh AppNavGraph, yang mengatur dua layar utama: FilmListScreen dan MovieDetailScreen. FilmListScreen menampilkan daftar film menggunakan LazyColumn, dan setiap film ditampilkan dalam kartu yang menyertakan tombol untuk membuka detail dan mengunjungi situs TMDB. Jika pengguna memilih film tertentu, aplikasi akan berpindah ke MovieDetailScreen, yang menampilkan informasi lengkap tentang film tersebut dan menyediakan tombol kembali ke daftar. Untuk pengelolaan preferensi pengguna, seperti mode gelap, aplikasi menyediakan kelas PreferenceManager yang memanfaatkan SharedPreferences.

Seluruh aplikasi mengandalkan ListFilmApp sebagai kelas Application untuk inisialisasi global, khususnya untuk mengaktifkan Timber sebagai logger. Dengan komponen-komponen ini, aplikasi ListFilm memiliki alur data yang terstruktur, dukungan caching lokal, navigasi yang jelas, serta desain UI modern yang didukung oleh Jetpack Compose. Aplikasi ini cukup lengkap untuk digunakan sebagai dasar pengembangan proyek menengah hingga lanjut.

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

https://github.com/rcl14/PrakMobile.git